Matthew Gardner

CS570

September 23rd, 2024

Tayyaba Shaheen

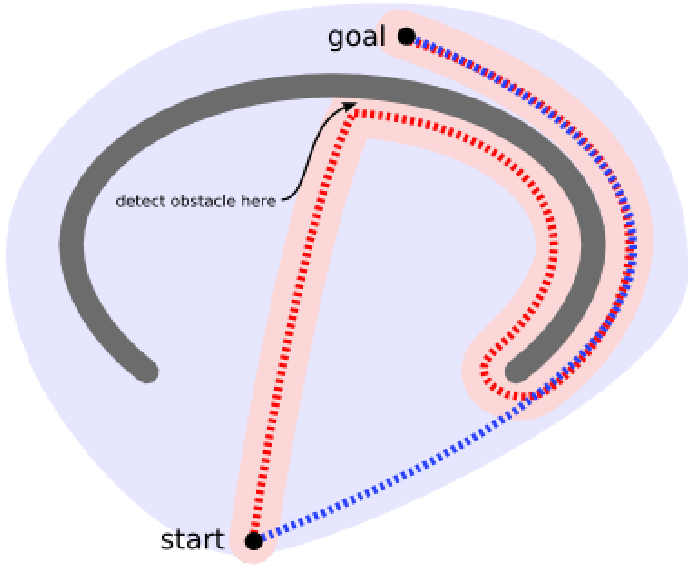
Artificial Intelligence in Video Games: A Literature Review

With the rapid advancements in Artificial Intelligence (AI), it is now integrated into nearly every piece of technology. One industry where AI has had a longstanding presence is the video game industry, where it has been used to enhance gameplay for decades. Alan Turing and Claude Shannon both used chess-playing programs as a theoretical route into the idea of how a computer could think back in the 50s. One of the earliest cases was from one of the biggest games, *Space Invaders*, in the year 1978. Although it wasn't true AI as we see it today, it was still a method of preprogrammed code to mimic random movement in the aliens. This primitive AI can be seen as a big reason for video games' rise and why the industry is so big today. In today's industry, AI has only become more sophisticated and important to the modern video game industry. This paper will examine various methods of AI implementation in video games by reviewing existing research and literature. As discussed in the project proposal, the two main methods investigated and discussed are the A* algorithm and machine learning. The research was also focused on AI used for pathfinding and problem-solving, relating again to the project proposal for building an AI to play the classic game *Snake*.

A quick note before beginning to introduce the ideas and methods found in the readings: all these methods require the first step to the project to be creating the game Snake in a controlled environment where you can manipulate variables and implement the specific Artificial Intelligence algorithm or method. To build this environment is rather simple. The first step would be to create an NxN grid of squares. At any time in the game, there should be an “apple” (just a colored-in square) that is the goal state of the snake. Once the snake connects with the apple, the length of the snake increases by one square behind it. The main goal of the game is to eat as many apples as possible. The only ways to die in the game are either by connecting with the border of the grid or connecting with the snake’s own body, which will grow as the game goes on.

As discussed in the project proposal the first idea of implementing the Snake game Artificial Intelligence was using the A* algorithm. The A* algorithm is one of the best and most used algorithms for path finding and graph traversals, so this method may not be the best for most video game AI situations it should work well for the snake game implementation discussed above as the entire snake game is basically just a graph traversal problem towards a goal. The main goal of the A* method implemented into this game is to find the quickest, safest, and most efficient path to the “apple” goal state and continue until no longer possible.

A pictorial example of the situation is displayed below, though it is not an exact relation to the *Snake* game, as *Snake* is entirely square based:



The situation above, however, is not the quickest path to the goal, as shown with the blue line. This is where the algorithm comes in, allowing the agent to discover obstacles before just reaching them. The A* algorithm is the most beneficial and efficient in these situations as it combines two other popular methods into one. The first part of the algorithm is based on Dijkstra's Algorithm, which visits vertices in the graph closest to the starting point, then repeats until reaching the end goal. This algorithm is guaranteed to find the shortest path to the goal but is not efficient as it looks at many different nodes on its search.

The second part of the A* method is the Greedy Best-First Search algorithm, which works similarly but uses heuristics (a heuristic is an estimate of how far from the goal any vertex is) to find the most efficient path to the goal. "Instead of selecting the vertex closest to the starting point, it selects the vertex closest to the goal. Greedy Best-First Search is not guaranteed to find the shortest path. However, it runs much quicker than Dijkstra's Algorithm because it uses the heuristic function to guide its way toward the goal very quickly" (Introduction to A*). By combining these two algorithms, you achieve an efficient runtime and the shortest route to the goal. In terms of the *Snake* game implementation, again, we can achieve an AI that uses this algorithm to get to the goal state until no longer possible.

he second method that will be discussed is a learning module, specifically the one most talked about in the research: a Convolutional Neural Network (CNN). Most of the readings, including one discussing exactly implementing a CNN to play Snake, stated that "our AI won't be perfect, and it won't fill in the entire map, but after some training, it will start playing at a level comparable with humans" (Ponteves, p.287). If you have not experienced endgame Snake, when there are very few empty spaces left on the board, there is a very specific technique you need to follow and set up earlier in the game to be able to fit your snake into the emptiest spaces.

To implement an AI learning model to accomplish this, the AI would have to learn to complete this specific path or have it preprogrammed when reaching a specific length. In this model, you create the learning model and allow it to just learn, letting it roam freely over the game. Take this example from the AI Crash Course: A Fun and Hands-on Introduction to Machine Learning book:

> “Think of it as teaching an AI to play with the actual buttons on a phone. If you keep trying to make your snake double back on itself when it's moving left, by pressing the go right button over and over again, the game will keep ignoring the impossible move you keep telling it to do, keep going left, and eventually crash. That's all the AI needs to learn.”

In more general terms, the AI has all the same information that a normal person playing the game would have and can do all the same things a normal person could. The main difference is the time it takes for the AI to learn how to play the game and achieve a higher score. A quick note on what differs between this type of learning AI and normal AI, like what was discussed in the previous section: “The world is consumed with the machine learning revolution, and particularly the search for a functional artificial general intelligence, or AGI. Not to be confused with a conscious AI, AGI is a broader definition of machine intelligence that seeks to apply generalized methods of learning and knowledge to a broad range of tasks, much like the ability we have with our brains.” (Lanham, p.8) The main difference is that the A* method is a much more methodical, mathematical approach, with heuristics and routing involved to find the fastest path. Whereas with a learning-based method, the AI just does what it wants every round, learning more and more about how to score higher points. From the reading *Hands-On Reinforcement Learning for Games*, there are four main elements to reward-based learning: the policy (representing the decisions and planning process of the agent), the reward function (the amount of reward an agent receives after completing a series of actions or an action), the value function (determining the value of a state over the long term), and finally the model (representing the environment in full, all game states). Now for the layout in terms of our *Snake* game: these four policies are easy to set. The policy is the possible directions the snake can change toward the goal or “apple.” The reward function is that the higher the score the snake achieves, the higher the reward it gets. The value function is again related to the score the snake has achieved. The model represents the entire board, such as all possible moves the snake could make and all the board squares available. Once this design and these policies are implemented, the AI just needs to be run and left to train itself to play the game. From the article read that did implement this AI into the *Snake* game, it took around 25,000 epochs for the snake to play at a level comparable with humans. As discussed at the start of this section, it is very unlikely that the snake will ever actually beat the game by filling every possible space. The more the AI is allowed to run, the better the snake will be, which is the main benefit of this method.

In conclusion, the readings that were conducted for this literature review did change my mind by a fair margin as I originally assumed that the A* method would be the easiest and best implementation. I could not find any A* method actual implementations and research so I may still be surprised when implemented, this lack of research done will probably be the focus of the actual project as there is a lack on the discussion. The main testing I plan on conducting for the research is comparing the A* and Learning models to each other to see which is first easier to implement, faster to start playing the game at the same level of a human, and finally the highest score achievable by each method. However, for now, I believe that the learning module will be the most successful and by far the most fun to implement, so I look forward to getting started on the actual implementation and design. Talking about implementation I was able to find a complete guide on creating learning model exactly for the game of snake, but I do not plan on copying exactly what they did but I will definity be taking notes on how they implemented it and combining it with other methods I found during the review.

Bibliography

Lanham Michael, Hands-On Reinforcement Learning for Games. 1st edition., [Erscheinungsort nicht ermittelbar] Packt Publishing 2020, 2020.

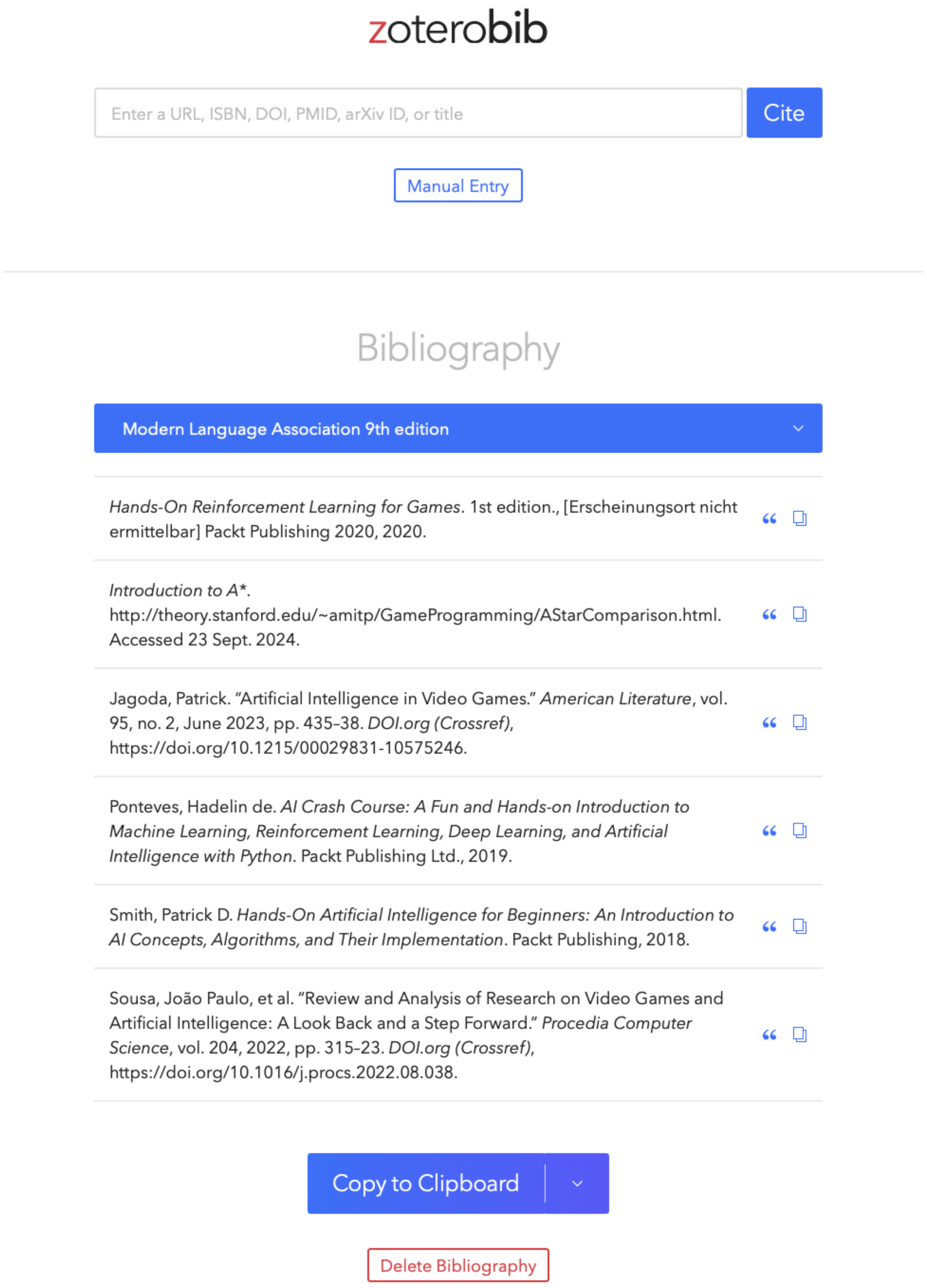
Introduction to A*. http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html. Accessed 23 Sept. 2024.

Jagoda, Patrick. "Artificial Intelligence in Video Games." *American Literature*, vol. 95, no. 2, June 2023, pp. 435–38. *DOI.org (Crossref)*, https://doi.org/10.1215/00029831-10575246.

Ponteves, Hadelin de. *AI Crash Course: A Fun and Hands-on Introduction to Machine Learning, Reinforcement Learning, Deep Learning, and Artificial Intelligence with Python*. Packt Publishing Ltd., 2019.

Smith, Patrick D. *Hands-On Artificial Intelligence for Beginners: An Introduction to AI Concepts, Algorithms, and Their Implementation*. Packt Publishing, 2018.

Sousa, João Paulo, et al. "Review and Analysis of Research on Video Games and Artificial Intelligence: A Look Back and a Step Forward." *Procedia Computer Science*, vol. 204, 2022, pp. 315–23. *DOI.org (Crossref)*, https://doi.org/10.1016/j.procs.2022.08.038.


zoterobib
Enter a URL, ISBN, DOI, PMID, arXiv ID, or title
Cite
Manual Entry

Bibliography

Modern Language Association 9th edition

Hands-On Reinforcement Learning for Games. 1st edition., [Erscheinungsort nicht ermittelbar] Packt Publishing 2020, 2020.

*Introduction to A**. http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html. Accessed 23 Sept. 2024.

Jagoda, Patrick. "Artificial Intelligence in Video Games." *American Literature*, vol. 95, no. 2, June 2023, pp. 435–38. *DOI.org (Crossref)*, https://doi.org/10.1215/00029831-10575246.

Ponteves, Hadelin de. *AI Crash Course: A Fun and Hands-on Introduction to Machine Learning, Reinforcement Learning, Deep Learning, and Artificial Intelligence with Python*. Packt Publishing Ltd., 2019.

Smith, Patrick D. *Hands-On Artificial Intelligence for Beginners: An Introduction to AI Concepts, Algorithms, and Their Implementation*. Packt Publishing, 2018.

Sousa, João Paulo, et al. "Review and Analysis of Research on Video Games and Artificial Intelligence: A Look Back and a Step Forward." *Procedia Computer Science*, vol. 204, 2022, pp. 315–23. *DOI.org (Crossref)*, https://doi.org/10.1016/j.procs.2022.08.038.